هندسة البرمجيات
Software Engineering

في ستة أيام......

تألـــــيف

أسماء المنقوش

مشرفة ساحة لغات البرمجة

APPROVED

التواصل مع القراء

إلى القارئ العزيز ،،،

حرصت الموسوعة العربية للكمبيوتر والإنترنت _ ومن منطلق اهتمامها العام بعلوم الحاسب والتقنية واهتمامها الخاص بتقديم هذه العلوم باللغة العربية _ على تقديم هذه السلسة من
الكتب الإلكترونية التى نتمنى أن تحقق طموحات القارئ العربى الذى اعتاد على قراءة أجود المطبوعات بكافة اللغات العالمية .

إن الموسوعة العربية _من خلال هذه السلسلة _ تطمح لتقديم سلسلة من الكتب بمستوى عالٍ من الجودة ، الشيء الذى لن يتحقق بدون ملاحظاتكم واقتراحاتكم حول السلسلة _ طريقة الكتابة ، الأخطاء الإملائية والنحوية ، التنظيم والترتيب ، طريقة نشر الكتاب وتوزيعه ، الإخراج الفنى ... الخ

ننتظر سماع أراءكم على البريد الإلكتروني المخصص لذلك
ebooks@c4arab.com
نرجو ذكر اسم الكتاب والكاتب والطبعة مع ذكر ملاحظاتكم لنا

تـــــــــهانى الســــــــــــبيت
مشرفة الموسوعة العربية للكمبيوتر والإنترنت

.. بســـــم الله الرحمـــــن الرحيـــــم ..

الدورات التعليمية .. هي مجموعة من الدورات التي تقدمها لكم الموسوعة العربية؛ بدأنا بتقديمها في الصيف تحت مسمى " الدورات الصيفية " وها هي تعود من جديد . حرصنا على تقديم دورات في مجالات مختلفة لنراعي أغلب الاهتمامات كما حرصنا على انتقاء الدورات المفيدة، غير المتكررة، بطريقة جادة تنقلك إلى الجو الدراسي في قاعات الجامعة و صفوف المعاهد و لكن في بيئة إلكترونية! كل هـذا مجــانـــا! ...
يوجد كذلك ساحة متخصصة لها ضمن مجموعة ساحات الموسوعة العربية للنقاش والأسئلة، تجدها **هنـــا**! ...





استفد واستثمر وقتك معنا! إذا كنت ترغب في تطوير ذاتك و توسيع نطاق ثقافتك في الحاسوب فاستغل كل دقيقة واستفد معنا! و لا تنسى أننا في عصر المعلومات والسرعة.

ابدأ الآن !انتقل لصفحة الدورات و اختر الدورة التي تناسبك، انتقل لصفحة الأساتذة للاطلاع على قائمة الأساتذة الّذين سيلقون المحاضرات ،انتقل لصفحة التسجيل كي تسجّل نفسك في إحدى الدورات، لن تستطيع المشاركة في أي دورة قبل أن تسجل. انتقل لصفحة المراجع كي تطلع على المراجع المقدمة من الأساتذة بخصوص الدورات الحالية .انتقل لصفحة الملتحقين لتطلع على بعض المعلومات عن الملتحقين في الدورات. انتقل لصفحة اتصل بنا كي ترسل لنا اقتراحاً أو طلباً. نحن بانتظارك! لكن الوقت محدود و عدد الملتحقين في كل دورة محدود لذا لا تتأخر في التسجيل من فضلك.

هذا الكتاب ....

ليس فى الأصل ألا دورة تم تدريسها فى ساحة الدورات التعليمية بالموسوعة العربية للكمبيوتر والإنترنت ، وتم جمع تلك الدروس وسلسلة النقاش التى دارت حولها هنا فى هذا الكتاب ، وتم وضع النقاشات على هيئة أسئلة وأجوبة لكى يستفيد الجميع منها ،،،،،،،،،

لذلك تعتبر <u>سلسلة كتب الدورات التعلمية</u> :

- **أول سلسلة كتاب إلكترونية عربية خاصة بالمبتدأين.**
- **السلسلة الوحيدة التى تتبع نظام الأسئلة والأجوبة الناتجة فعلاً من مشاكل حقيقة لأشخاص من مختلف الأماكن والدول ، مما يهيئ عندك نوع من استعداد لأى مشكلة وكيفية التعامل معها.**
- **تعتبر سلسلة الكتاب الوحيدة المدعومة اربع وعشرين ساعة طوال العام ، فيمكنك الاستفسار عن اى مشكلة وحلها عن طريق وضعها فى ساحة النقاش والأسئلة بالموسوعة .**
- **إن هذا الكتاب هو من اجل نشر المعرفة وتوسيع التفكير المنطقى الأساسي ، الاحتراف هو ليس الهدف فى حد ذاته، بل الاستطلاع واكتشاف الذات والإلمام الجيد بالأساسيات والمبادئ الأولية من اجل شق طريق النجاح بكل سهولة ويسر.**

المحتويات ..

بسم الله الرحمن الرحيم

الدرس الأول: ماذا نعني بهندسة البرمجيات؟

أهداف الدرس الأول:
سوف نحاول خلال هذا الدرس الإجابة على هذه الأسئلة:

- ما هي هندسة البرمجيات؟
- من يشارك بها؟
- ما هي مكونات النظم البرمجية؟
- وكيف يتم بنائها؟

مقدمة:
لم يعد خافيا على أي منا أهمية البرمجيات Software في حياتنا اليومية سواء في البيت أو المصنع أو المستشفى أو ... الخ، فنحن نتعامل يوميا مع العديد من الأجهزة والمعدات التي تعتمد في عملها على البرمجيات ومن المهم لنا أن تعمل هذه الأجهزة وبرامجها بالشكل والكفاءة التي نتوقعها منها. لذا فإن هندسة البرمجيات أصبحت اليوم أكثر أهمية من أي وقت مضى.

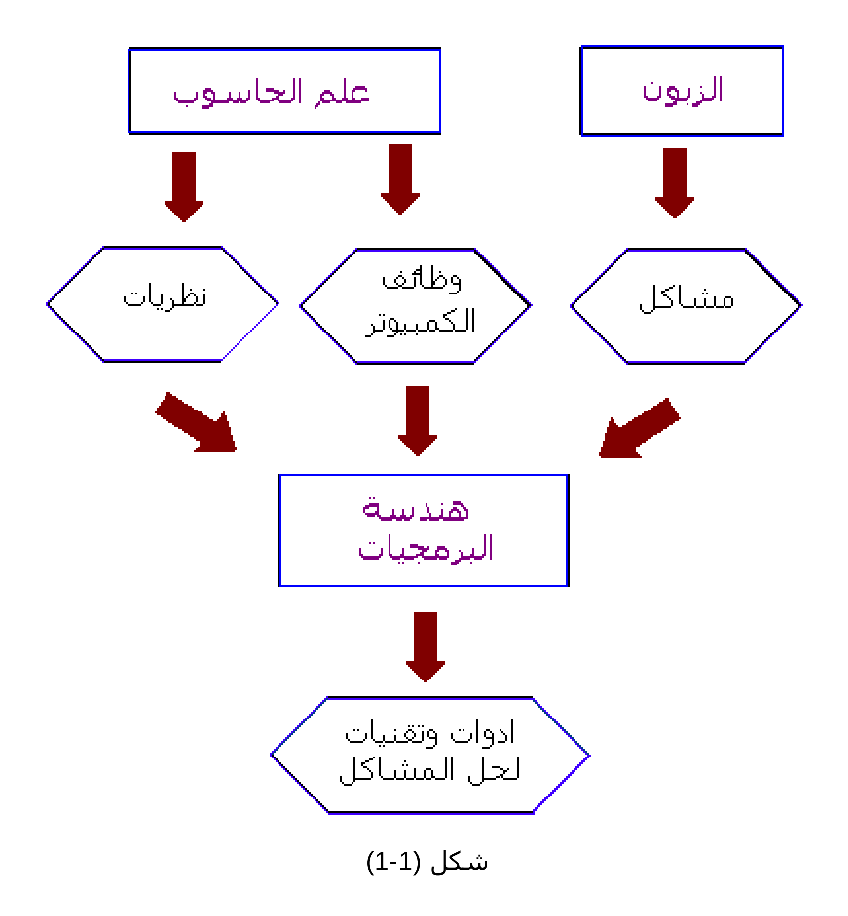
المرجع:

1- Shari Pfleeger, "Software Engineering - Theory and Practice", 2nd Edition

ما هي هندسة البرمجيات؟
لنفهم معا علاقة هندسة البرمجيات بعلوم الكومبيوتر، دعونا نأخذ هذا المثال عن علم الكيمياء واستخدامه في حل المشاكل التي نقابلها في حياتنا اليومية.
يهتم الكيميائي بدراسة المواد الكيميائية (تركيبها، تفاعلاتها، والنظريات التي تحكم سلوكها).
بينما المهندس الكيميائي يستخدم النتائج التي توصل إليها الكيميائي لحل المشاكل التي يطلب منه إيجاد حل لها.
من وجهه نظر الكيميائي الكمياء هي **موضوع الدراسة بحد ذاتها**.
ومن وجهه نظر المهندس الكيميائي الكيمياء هي **أداة** tool **تستخدم لأيجاد الحلول** لمشاكل عامة) وقد لا تكون هذه المشكلة ذات طبيعة كيميائية بحد ذاتها.(

وبنفس الفكرة يمكن النظر إلى علم الحوسبة computer science حيث يكون تركيزنا على الحواسيب ولغات البرمجة لدرستها وتطويرها في حد ذاتها.
أو يمكن النظر إليها والتعامل بها على أنها أدوات نستخدمها عند تصميم وتطوير حل لمشكلة ما تواجهنا أو الآخرين.

مهندس البرمجيات Software Engineer يعتبر أن الكمبيوتر هو أداة لحل المشاكل.problem-solving tool
وعليه أن يستخدم معلوماته حول الحاسوب وعلم الحوسبة للمساعدة في حل المشكلة التي يطلب منه إيجاد حل لها.



شكل (1-1)

ولكن ومن المهم أن نتذكر أن عملية كتابة البرامج تعد **فن** Art بقدر ما هي **علم**، لماذا؟

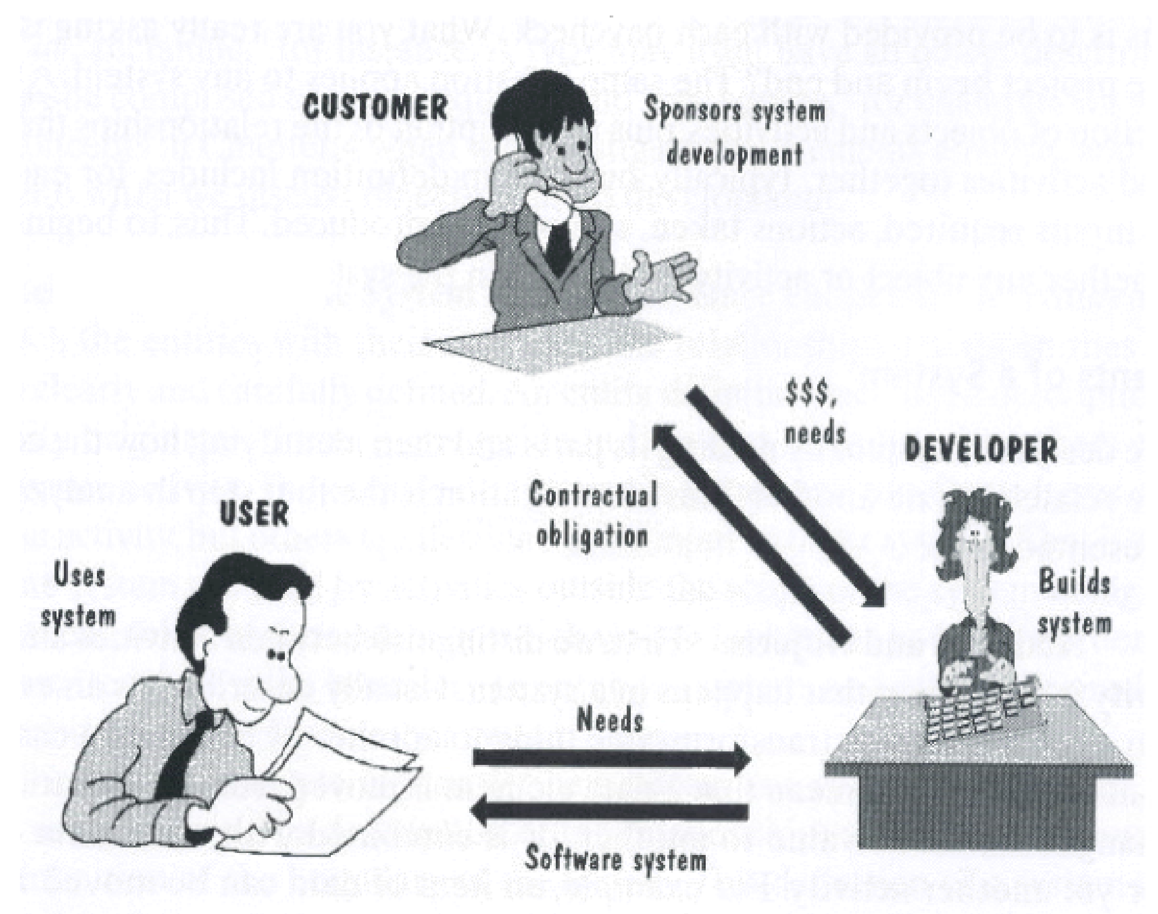
لأنه يمكن لأي شخص لديه معرفة كافية بأحد لغات برمجة الحاسوب hacker أن يكتب برنامج ليؤدي مهمة محددة، لكن الامر يتطلب مهارة ومعرفة مهندس برمجيات محترف لكتابة برنامج أكثر تناسقا ووضوحا ،وأسهل في الصيانة، ويقوم بالمهمة المطلوبة منه بفعالية ودقة أكبر.

أي أن، **هندسة البرمجيات تعنى بتصميم وتطوير برامج ذات جودة عالية**.

من يشارك في هذه العملية؟
المشاركون في عملية صناعة البرنامج، عادة ما يندرجون تحت ثلاث مجموعات:

- الزبون Customer: وهو الشركة (أو الشخص) الممولة لمشر وع تطوير البرنامج المطلوب
- المستخدم User: الشخص (أو مجموعة الاشخاص) الذي سوف يقوم فعلا باستعمال البرنامج، والتعامل معه مباشرة.
- المطور Developer: وهو الشركة (أو الشخص) الذي سوف يقوم بتطوير البرنامج لصالح الزبون.

الشكل التالي يظهر العلاقة بين الفئات الثلاثة السابقة



شكل (1-2)

مكونات النظام

مشاريعنا التي نطورها لن تعمل في الفراغ، فعليها أن تتفاعل مع مستخدمين، أجهزة ومعدات متنوعة، نظم تشغيل وبرامج وملفات وقواعد بيانات .... إلخ و ربما حتى أنظمة حواسيب آخرى. لهذا يجب تعريف حدود النظام ومكوناته جيدا .أي يجب تعريف ما الذي يشتمل عليه النظام وما الذي لا يشتمل عليه.

أي نظام هو عبارة عن مجموعة من الكائنات objects والنشاطات activities بالإضافة إلى وصف للعلاقات التي تربط تلك الكائنات والنشاطات معا. مع تعريف قائمة المدخلات المطلوبة والخطوات المتبعة والمخرجات الناتجة لكل نشاط.

أول خطوات تحليل المشكلة هو فهم ماهية المشكلة وتعريفها بوضوح، لذا علينا أولا أن نصف النظام بتحديد مكوناته والعلاقات التي تربط بين هذه المكونات.
.1النشاطات والكائنات: النشاط هو عملية تحدث بالنظام وعادة ما يوصف كحدث يتم من خلال حافز. النشاط يغير شئ ما إلى آخر بتغير خواصه (صفاته(
هذا التغير يمكن أن يعنى تحويل أحد عناصر البيانات من موقع إلى آخر، أو تعديل قيمته إلى قيمة مختلفة. هذه العناصر تسمى كائنات objects وهي عادة ماتكون مرتبطة ببعضها البعض بشكل أو بأخر. مثلا الكائنات يمكن أن تكون مرتبة في مصفوفة أو سجل) قيد.(

وصف هذه الكائنات نوعها، النشاطات التي يمكن إجرائها عليها ... يجب وضعها بدقة هي ايضا.

2.العلاقات وحدود النظامRelationships and System Boundary
بعد تعريف الكائنات والنشاطات جيدا، يمكن أن نربط بين كل كائن والنشاطات المتعلقة به بدقة. تعريف الكائن يتضمن الموقع الذي سوف ينشأ به(نعض العناصر يمكن أن تكون موجودة بملف سبق انشاءه، والبعض قد يتم انشاءه خلال حدث ما(، والهدف من انشاءه(بعض الكائنات تستخدم من قبل نشاط واحد فقط والبعض يمكن أن يستعمل من قبل نظم آخرى كمدخلات , Input) لذا يمكن أن نعتبر أن لنظامنا حدود boundary بعض الكائنات بمكن أن تعبر هذه الحدود إلى داخل النظام، والبعض الآخر هي مخرجات من نظامنا ويمكن أن ترحل إلى نظم آخرى.

بهذا يمكن أن نعرف النظام A System على أنه تجمع من:
·مجموعة من الكائنات.entities
·مجموعة من الانشطة.activities
·وصف للعلاقات بين الكائنات والانشطة.Relationship
·تعريف لحدود النظام.boundary

كيف نبي نظام؟

إذا طلب منا عميل تطوير نظام (برنامج) له، لحل مشكلة معينة تواجهه في عمله. فمثلا يحتاج نظام حماية لشركته، أو نظام صرف آلي لبنك، أو ممكن أن يكون صاحب مكتبة أو متجر و يريد تغير نظام البيع و الشراء أو العرض ليتم بشكل آلي. علينا اتباع الخطوات التالية لبناء هذا النظام:
1.عقد اجتماع مع العميل لتحديد متطلباته، هذه المتطلبات تشمل وصف النظام بجميع مكوناته التي شرحنا.
2.وضع تصميم عام للنظام يحقق المتطلبات التي حددها العميل، وعرضه على العميل ليوضح له الشكل الذي سيظهر عليه النظام عند الانتهاء، و ومراجعته معه لأخذ موافقته عليه.
3.بعد موافقة العميل على التصميم يتم العمل على وضع التصاميم التفصيلية لأجزاء المشروع.
4.كتابة البرنامج
5.اختباره، واعادة مراجعة المتطلبات التي وضعها العميل للتأكد من تحققها في البرنامج.
6.تسليم النظام إلى العميل.
7.بعد تسلم العميل للنظام قد تظهر بعض المشاكل أو الاخطاء التي لم تظهر خلال عملية الاختبار، والتي تجب على المطور اصلاحها فيما يعرف بصيانة النظام.

خلال الدروس التالية من الدورة سنتعرف على كل خطوة من هذه الخطوات وكيف تتم بشكل مبسط، وسوف نخوض في مزيد من التفاصيل في دروس لاحقة بإذن الله.

(•.¸¸.•´¯`•.نها ية الدرس الأول•.¸¸.•´¯`•.)

(●·.·´¯`·.·●نقاش الدرس الأول●·.·´¯`·.·●)

س1 - هل المقصود بهذي الجملة ان المبرمج لا يستطيع حل المشكله فقط مهندس البرمجيات هو الذي يستطيع؟؟؟؟

ممكن أن يوجد شخص تعلم البرمجة دون أن يدرس هندسة برمجيات و شخص آخر درس هندسة البرمجيات وبالطبع علوم الحاسوب .. لو اعطيت هذين الشخصين مشكلة ما .. سيكون حل مهندس البرمجيات للمشكلة أفضل من حل المبرمج الذي لم يدرس هندسة البرمجيات

"تستطيع أن تقول أن كل مهندس برمجيات هو مبرمج بينما ليس كل مبرمج هو مهندس برمجيات"

نعم هذا هو المقصود، هندسة البرمجات لا تهتم فقط بكتابة برنامج يؤدي مهمة محددة فحسب، بل أنها تهتم بما هو أكثر من ذلك "جودة البرنامج"

كلمة مبرمج أو Hacker تطلق على كل من يعرف كيف يكتب برنامج للقيام بأداء عمل ما..
ولكن كلمة مهندس برمجيات لا تطلق إلا على من يكتب هذه البرمجيات باسلوب علمي يسعى من خلاله إلى أن تكون برامجه ذات جودة عالية.

س2 - ما المقصود في فن Art **؟**

المقصود بكلمة Art هو الفن .. لأن كلمة الفن Art = بالانجليزي

واما المقصود بالدرس ..
هو ان البرمجة فن وتذوق اكثر من ان تكون علم فقط أي انه يمكن كتابة نفس البرنامج باسلوب مختلف من شخصيين مختلفيين ويودي نفس المهام...
وهذا كله يعتمد علي اسلوب المبرمج وكيفية حله للمشكلة وطريقة صيغته للبرنامج .

س3 - هل يوجد فرق بين مهندس برمجيات و محلل نظم ؟

نعم هناك فرق بين مهندس البرمجيات ومحلل النظم فمثلا في الدول المتقدمة يقوم محلل النظام بدراسة المشروع المراد تنفيذه وكيفية حل المشاكل التي توجه كما ويقوم بدراسة الجدوى ومعرفة متطلبات النظام ...الخ

أي انه يقوم بتحليل النظام المراد بنائه تحليل دقيق .

اما مهندس البرمجيات فيقوم ببرمجة النظام وتهيئته كي يظهر في الصورة النهائية..
أي يحتاج على الاقل الي شخصين كي يتم بناء النظام او البرنامج المطلوب.

الدرس الثاني: دورة حياة تطوير المشروع

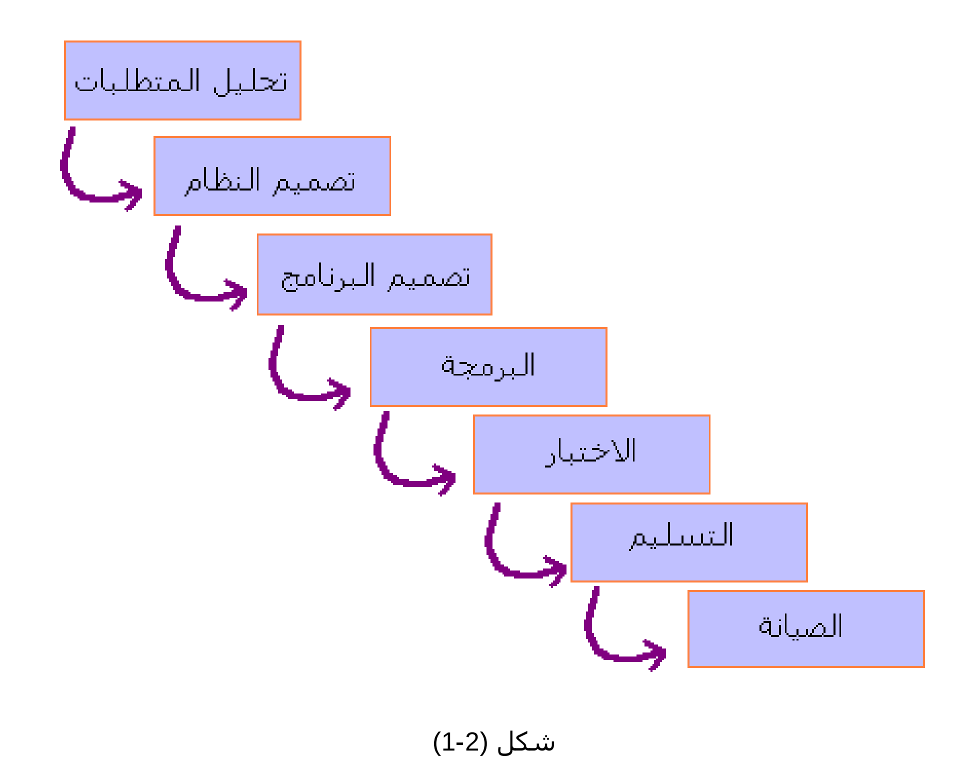
أهداف الدرس الثاني:
كما رأينا في الدرس الاول فإن هندسة البرمجيات هو عمل إبداعي يتم إداءه خطوة بخطوة، ويتعاون فيه عدد من الاشخاص لكل منهم مهمة محددة .في هذا الدرس سوف نناقش الخطوات التي يتم اتباعها عند تطوير مشروع برمجي بمزيد من التفاصيل ونبحث في الطرق المستخدمة لتنظيم هذا العمل (صناعة البرمجيات(

مقدمة:
عملية بناء أي منتج تمر بعدة مراحل يطلق عليها عادة "دورة الحياةLife Cycle "، ومما تعلمنا في الدرس السابق فإن دروة حياة تطوير أي نظام برمجي Software development life cycle تتضمن المراحل التالية:
1.تحديد وتعريف المتطلباتRequirements analysis and definition
2.تصميم النظامSystem design
3.تصميم البرنامجProgram design
4.كتابة البرنامج (تطويره)Program implementation (
5.أختبار وحدات البرنامجUnit testing
6.أختبار النظامsystem testing
7.تسليم النظامsystem delivery
8.الصيانةmaintenance

كل مرحلة من تلك المراحل تتضمن العديد من الخطوات أو النشاطات ولكل منها مدخلاتها ومخرجاتها وتأثرها على جودة المنتج النهائي) البرنامج.(
دورة حياة أي منتج تبدأ بأول خطوة وهي تحديد المتطلبات وتتدرج إلى باقي الخطوات كما هي مرتبة حتى الوصول إلى آخر خطوة وهي تسليم البرنامج وصيانته (إن دعت الحاجة)، إلا أن التجارب العملية تظهر أن هذا ليس ضروريا وأن دورة حياة تطوير البرامج قد تأخذ أشكال (أو أنماط) مختلفة. وفي هذا الدرس سوف نتعرف إلى هذه الأنماط

أنماط دورة الحياة:Lifecycle Models

النموذج الانحداري Waterfall Model
في هذا النموذج تسير دورة الحياة بشكل تدريجي بدأ من الخطوة (1) وحتى الخطوة (8)، وكما يظهر بالشكل (1) فإن كل مرحلة تبدأ بعد الأنتهاء من المرحلة التي تسبقها مباشرة.



شكل (2-1)

يتميز النموذج الانحداري بالبساطة، ولذا فإنه يسّهل على المطور توضيح كيفية سير العمل بالمشروع للعميل (الذي عادة لا يعرف الكثير عن صنع البرمجيات) والمراحل المتبقية من العمل. وقد كان هذا النموذج أساس عمل كثير من المؤسسات لفترة طويلة مثل وزارة الدفاع الامريكية، واستنبط منه العديد من النماذج الاكثر تعقيدا.
إلا أن لهذا النموذج العديد من العيوب، أهمها أنه لا يعكس الطريقة التي يعمل بها المطورون في الواقع. فباستثناء المشاريع الصغيرة والبسيطة (أي أنها مفهومة بشكل جيد للمطور) فإن البرمجيات عادة ما تنتج بعد قدر هائل من التكرار والاعادة. في حين أن هذا النموذج يفترض أن يكون الحل واضح ومفهوم وسبق تحليله بالكامل قبل مباشرة مرحلة التصميم وهو أمر يكاد يكون شبه مستحيل مع الانظمة الضخمة. وحتى إن كان ممكن فإنه يأخذ وقت طويل جدا (ربما سنوات!)

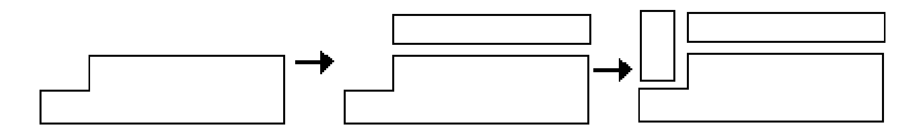
باختصار،النموذج الانحداري سهل الفهم و بسيط في إدارته. لكن مميزاته تبدأ في التداعي بمجرد أن يزداد تعقيد المشروع.

التطوير على مراحل Phased Development
حسب النموذج الانحداري فإنه يجب على المطورين إنهاء مرحلة تحليل المشروع بشكل تام قبل البدأ في التصميم، وكما وضحنا فإن هذه المرحلة قد تتطلب وقت طويل في بعض المشاريع وقد تمر عدة سنوات قبل أن يرى البرنامج النهائي النور، ولكن هل يمكن لسوق العمل الانتظار كل هذا الوقت؟!

الاجابة بالطبع لا.
لذا كان لابد من ايجاد طرق آخرى لتقليل زمن تطوير المشروع Cycle time. أحد هذه الطرق هي التطوير على مراحل Phased Development حيث يتم تطوير النظام على عدة مراحل، بتقديم إصدار من البرنامج به بعض الوظائف للعميل والعمل على تطوير الاصدار الاحق الذي سوف يقدم له بقية الوظائف.

يوجد عدة طرق يمكن بها تنظيم عملية تطوير إصدارات البرنامج، ومن اشهرها:
· النموذج التزايدي Incremental model

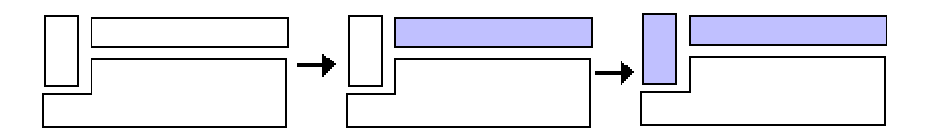
حيث يتم تقسيم النظام المطلوب تطويره إلى عدة اجزاء حسب الوظائف التي يعتين عليه القيام بها، يبدأ أول إصدار بأحد تلك الاجزاء ومع الوقت يتم إضافة المزيد من الاجزاء (الوظائف) حتى يتم الانتهاء من تطوير النظام بشكل تام وحسب متطلبات العميل.



Incremental Development

٠النموذج التكراريIterative model

هذه المرة يتم تسليم برنامج بكامل الوظائف من أول مرة، ولكن يتم تعديل وتغيير بعض تلك الوظائف مع كل إصدار من البرنامج.



Iterative Development

من مميزات هذا الأسلوب أنه يمكن المطورين من الحصول على ملاحظات وتقييم الزبون مبكرا و بصورة منتظمة، ورصد الصعوبات المحتملة قبل التمادي بعيدا في عمليات التطوير.كم أنه يمّكن من اكتشاف مدى حجم و تعقيد العمل مبكرا.

النموذج اللولبي Spiral Model

وهو شبيه لدرجة كبيرة إلى النموذج التزايدي والتكراري، ولكن فيه يتم دمج فعاليات التطوير مع إدارة المخاطر riskمن إجل التحكم بها وتقليلها.
يبدأ النموذج اللولبي بمتطلبات العميل مع خطة العمل المبدئية (الميزانية، قيود النظام، والبدائل المتاحة). ثم يتقدم خطوة إلى الامام بتقدير المخاطر وتمثيل البدائل المتاحة قبل تقديم ما يعرف بـ "وثيقة العمليات " Concept of Operationsالتي تصف وبشكل عام (بدون الدخول في التفاصيل) كيف يجب على النظام أن يعمل. بعدها يتم تحديد وتدقيق المتطلبات للتأكد من أنها تامة ودقيقة إلى أقصى حد ممكن.
بذلك تكون وثيقة العمليات هي المنتج من الطور الأول، و المتطلبات هي المنتج الاساسي من الطور الثاني.
وفي الطور الثالث تتم عملية التصميم، أما الاختبار فيتم خلال الطور الرابع.
في كل طور أو مرحلة يساعد تحليل المخاطر على تقدير البدائل المختلفة في ضوء متطلبات وقيود النظام، وتساعد النمذجة على التحقق من ملائمة أي بديل قبل أعتماده.

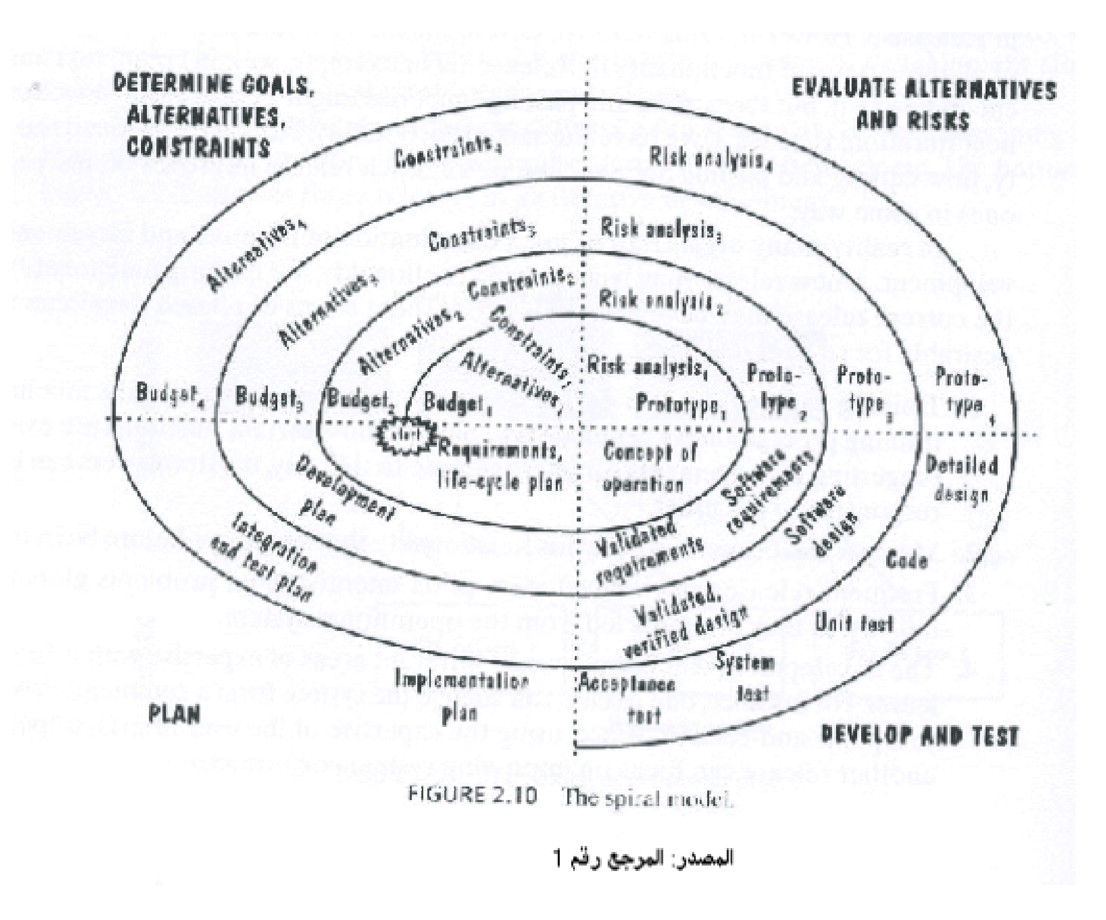


FIGURE 2.10 The spiral model.

المصدر: المرجع رقم 1

شكل (2-2)

(•·.·´¯`·.·•نهاية الدرس الثاني•·.·´¯`·.·•)

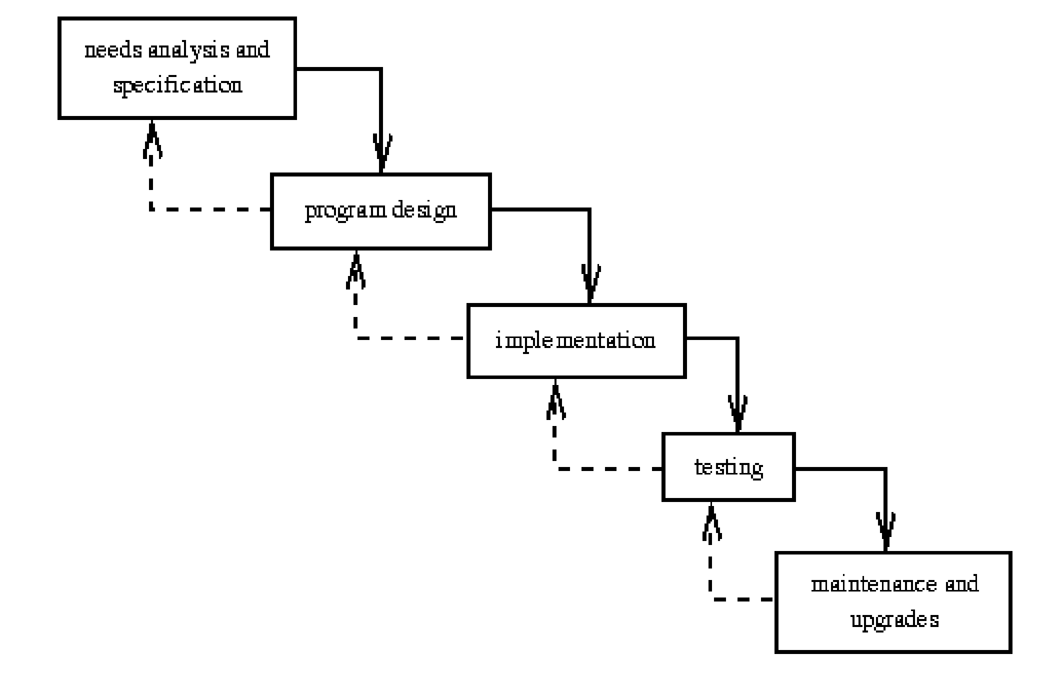
) ●¸.·´¯`·.¸●نقاش الدرس الثاني●¸.·´¯`·.¸● (

لي ملاحظه و هي في الـ.. WaterFall Model
هل من الممكن أن يكون هناك back track من Phase لأخر و كلما كان هذا الرجوع أكبر كلما كان مكلف أكثر من ناحية الوقت و التعديل و المال ؟؟

نعم من الممكن أن يكون هناك back track من Phase
وغالبا ما يكون هذا ما يحصل بالفعل عند التطبيق العملي...

الـ backtrack ممكن يحدث في جميع الـ models وليس هذا فقط .. فبعد كل مرحلة ممكن يُكتشف أن ناتج أحد المراحل السابقة لم يكن صحيح ويحتاج إلى تعديل
وهذا ما يجعل أنماط آخرى كالنمط اللولبي أكثر تفضيلا.

صورة للـ backtrack بالنسبة للـWaterfall Model



الدرس الثالث: دراسة المتطلبات

في هذا الدرس سوف نبدأ في دراسة أول (ولعلها أهم) خطوة في تطوير البرامج وهي تحديد متطلبات النظام .Capturing the requirements

الهدف من تحديد المتطلبات هو فهم ما يتوقعه العميل والمستخدم من النظام (ما الذي يمكن للنظام أداؤه وما لا يمكنه أداؤه).فقد يكون النظام المطلوب تصميمه بديل لنظام أو لطريقة مستخدمة لأداء مهمة محددة، أو ممكن أن يكون نظام جديد يقدم خدمة جديدة لم يسبق تقديمها من قبل. فلكل نظام برمجي وظيفة معينة، تحدد بما يمكن له أن يقوم به من أجل أداء تلك الوظيفة.

المتطلبات :هي تعريف لشكل النظام أو وصف لما يستطيع هذا النظام أن يقوم به لأداء وظيفته التي سيصمم من أجلها.

خطوات تحديد المتطلبات :

أولا: الاجتماع مع العميل للتعرف على المتطلبات:
وهذه خطوة هامة جدا إذ أن بقية الخطوات التالية تعتمد عليها بشكل أساسي. لذا يجب علينا أن نستخدم كافة التقنيات المتاحة لنكتشف ما الذي يطلبه العميل والمستخدم، نبدأ بفهم وتحليل المشكلة التي تواجه المستخدم بكل أبعادها، نتعرف على العمليات والمصادر التي تتضمنها المشكلة والعلاقات التي تربطها معا و نحدد حدود النظام. وهذا يمكن أن يتم من خلال:

- طرح الأسئلة على العميل، ومن المفيد أحيانا أن نطرح نفس السؤال ولكن بأسلوب مختلف أكثر من مرة فهذا يساعدنا على التأكد من أننا نفهم ما يقصده العميل بالتحديد.
- عرض نظم مشابه للنظام المطلوب سبق تصميمها من قبل.
- تصميم وعرض نماذج لأجزاء من النظام المطلوب أو للنظام بالكامل.

تقسم المتطلبات إلى عدة عناصر تشمل:

- البيئة المحيطة بالنظامPhysical Environment
- وجهات الاستخدامInterfaces
- المستخدمين وإمكاناتهمUsers and human factors
- وظائف النظامFunctionality
- التوثيقDocumentation
- البياناتData
- المصادرResources
- الأمنSecurity
- ضمان الجودةQuality Assurance

ويجب التأكد من أن نناقش جميع هذه العناصر

ثانيا: تسجيل هذه المتطلبات في وثائق أو قاعدة بيانات، وعرضها على العميل ليوافق عليها باعتبار أنها ما يطلبه بالفعل
المتطلبات لا تصف فقط تدفق البيانات والمعلومات من وإلى النظام، وأما تصف كذلك القيود المفروضة على عمل النظام. وبذلك فإن عملية تحديد المتطلبات تخدم ثلاثة أغراض:

- أولا تمكن المطورين من شرح فهمهم للطريقة التي يود المستخدمين أن يعمل بها النظام.
- ثانيا توضح للمصممين ماهية الوظائف والخصائص التي سيمتاز بها النظام ,
- وثالثا: توضح المتطلبات لفريق الاختبار ما الذي يجب إثباته لإقناع الزبون أن النظام الذي تم تطويره هو ما سبق أن طلبه بالضبط .

لذلك ولضمان أن كلا من المطورين والزبون متفاهمون تماما على ما يجب القيام به، فإن المتطلبات المسجلة حتى هذه الخطوة يجب أن تكون لها الصفات التالية:
1.أن تكون صحيحة Correct وخالية من الأخطاء.
2.أن تكون ثابتة consistent بمعنى أن لا يكون هناك أي تعارض بين متطلب وآخر.
3.أن تكون تامة Complete يجب أن يتم ذكر جميع الحالات المحتملة للنظام، المدخلات، المخرجات المتوقعة منه، ...الخ .
4.أن تكون واقعية realistic بمعنى أن تكون قابلة للتطبيق في الواقع.
5.أن تكون متعلقة بأمور ضرورة للعميل، ويتطلبها النظام.
6.أن يكون من الممكن التحقق منهاverifiable
7.أن تكون قابلة للتتبعtraceable
يطلق على هذه الوثائق "وثائق تعريف المتطلبات Requirement Definition Document "

ثالثا: إعادة تسجيل المتطلبات بشكل رياضي mathematical ليقوم المصممون بتحويل تلك المتطلبات إلى تصميم جيد للنظام في مرحلة التصميم.

لسنوات عديدة كان يتم الاكتفاء بوثيقة تعريف المتطلبات (التي تحدثنا عنها قبل قليل) والتي تكتب باستعمال اللغة الطبيعية) لغة البشر) لوصف وتسجيل متطلبات النظم بحيث يمكن للعميل أن يفهم كل كلمة موجودة بها، إلا أن ذلك يسبب العديد من المشاكل والتي يعود سببها في أغلب الأحيان إلى سوء تفسير بعض التعبيرات للمستخدمين من قبل المصمم أو العكس، فعلى سبيل المثال قد يطلق المستخدم على النظام التعبير (متوقف عن العمل) إذا كان النظام مشغول بعملية تسجيل احتياطي backup باعتبار أن لا يستجيب لأوامر المستخدم في هذه الحالة، بينما يعتبر المصمم أن النظام في هذه الحالة (مستمر في العمل) لأنه يقوم بمهمة أساسية!
لذا فأن الاعتماد على اللغة البشرية بشكل تام قد يؤدي إلى أخطاء كثيرة عند تصميم النظام، وينتج عنها نظام لا يقبله العميل لأنه لا يلبي متطلباته التي حددها من قبل، لذلك يتم كتابة نوع ثاني من الوثائق تسمى "وثائق مواصفات المتطلبات Requirement specification Document "وهي تكتب باستعمال وسائل وطرق خاصة ابتكرها مهندسو البرمجيات لكتابة المتطلبات باسلوب تقني بحت. منها على سبيل المثال: لغة النمذجة الموحدة UML Unified Modeling Language و هي لغة نمذجة رسومية تقدم لنا صيغة لوصف العناصر الرئيسية للنظم البرمجية.
الشكل التالي يعرض مثال على استعمالUML

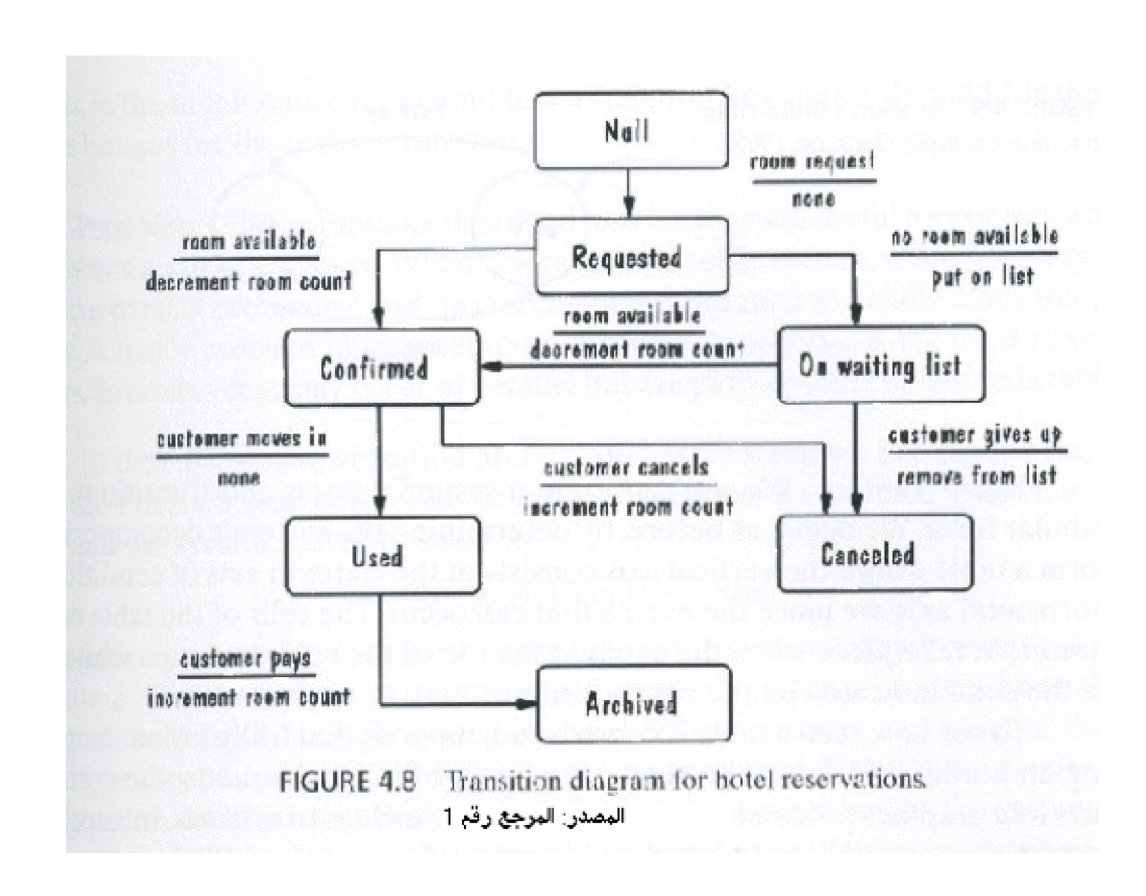


FIGURE 4.8 Transition diagram for hotel reservations

المصدر: المرجع رقم 1

رابعا: التثبت والتحقق من المتطلبات التي تم تسجليها في كلا من وثيقة تعريف المتطلبات (والتي تقدم للعميل) ووثيقة مواصفات المتطلبات (والتي تقدم للمصمم (للتأكد من صحتهما وشموليتهما وأن كلا منهما لا تعارض الثانية في أي نقطة، وإلا فإن النتيجة سوف تكون نظام لا يلبي طلبات العميل.!

(•.¸¸.•´¯`•.¸¸.•نهاية الدرس الثالث•.¸¸.•´¯`•.¸¸.•)

) ●·.·´¯`·.·●نقاش الدرس الثالث●·.·´¯`·.·● (

ــــــــــــــــــــــــــــــــــــأقتباســـــــــــــــــــــــــــــــــــ

أولا: الاجتماع مع العميل للتعرف على المتطلبات:
وهذه خطوة هامة جدا إذ أن بقية الخطوات التالية تعتمد عليها بشكل أساسي. لذا يجب علينا أن نستخدم كافة التقنيات المتاحة لنكتشف ما الذي يطلبه العميل والمستخدم،............ الخ
اخر شي قلتي" : ويجب التأكد من أن نناقش جميع هذه العناصر"

ـــ

هل تقصدين النقاش مع العميل!!؟

نعم يجب مناقشة كل نقطة مع العميل للتأكد من اننا فهمنا ما يقصده تماما.

البيئة المحيطة بالنظام.. Physical Environment
لم أفهم ما تقصدين بالبيئة المحيطة؟؟

يقصد بها كل ما يحيط بالنظام وليس من مكوناته مثلا الموقع الذي سيعمل به النظام، هل هو ثابت في موقع واحد أكثر أو يمكن أن يتم نقله إلى مواقع مختلفة (طبعا الحديث يشمل النظام كامل + Hardware software)

نريد توضيح او مثال عن صفات المتطلبات الأتية :

1. **أن يكون من الممكن التحقق منها** verifiable

بمعنى أن تكتب المتطلبات بحيث تكون قابلة للاختبار للتأكد من تحققت، فمثلا قد يذكر العميل أن يريد من النظام أن يكون ذا استجابة سريعة!
ما مقدار السرعة المطلوب؟
قد يرى المصمم أن الانتظار لمدة 50 ثانية مناسب كحد أقصى، بينما يتوقع الزبون زمن انتظار 20 ثانية كحد أقصى!

2. **أن تكون قابلة للتتبع** traceable

بمعنى أن تكون المتطلبات مكتوبة بحيث يسهل تتبعها للتأكد من أن كل وظيفة مطلوبة من النظام تم استيفائها من خلال المتطلبات.

الدرس الرابع: تصميم النظام

نكمل مع خطوات بناء النظام، وهذه المرة سوف نتحدث عن خطوة "تصميم النظام "

ما هو التصميم؟

التصميم هو عملية إبداعية لإيجاد حل لمشكلة، كما تطلق عادة كلمة تصميم على وصف هذا الحل. حيث نستفيد من المتطلبات التي حددنها في الخطوة السابقة في التعرف على المشكلة، ثم نبدأ في التفكير في الحل الذي يفي بجميع الشروط والمواصفات التي تحددها المتطلبات، وغالبا ما يمكن إيجاد عدد غير محدود من الحلول يمكن لنا أن نختار أحدها و الذي نجده الأنسب من بينها.

عند الانتهاء من خطوة تحديد المتطلبات، فإننا ننتهي بوثيقتين (كما ذكرنا في الدرس السابق) الأولى هي (وثيقة تعريف المتطلبات) ويتم تقديمها للعميل والثانية (وثيقة مواصفات المتطلبات) ويتم تقديمها للمصمم.

ودور المصمم هو تحويل هذه الوثائق إلى نظام يرضي العميل (يلبي احتياجاته)، وفي نفس الوقت يرضي المطور (يمكن تطبيقه.(
لذا فإن عملية التصميم في عملية تكرارية iterative من خطواتين:

أولا: يتم إنتاج التصميم التصوري conceptual design والذي يوضح للعميل ما الذي سيقوم به النظام بالتحديد .
وفي حال موافقة العميل على هذا النظام، يتم الانتقال للخطوة التالية.

ثانيا: تحويل التصميم التصوري إلى وثيقة بها تفاصيل أكثر عن التصميم يطلق عليها اسم التصميم التقني technical design والذي يجب أن يظهر للمطور ما هي المعدات والبرمجيات اللازمة لبناء النظام.

أحيانا يتطلب الأمر للعودة إلى الخطوة الأولى) التصميم التصوري) والتعديل عليه، لذا فأنها عملية تكرارية حتى الوصول إلى التصميم الذي يرضي العميل ويمكن تطبيقه على أرض الواقع في ظل الإمكانيات المتاحة للمطورين.

التصميم التصوري: conceptual design

يركز هذا التصميم على وظائف النظام functions ويكتب بلغة يمكن للعميل أن يفهمها (لغة البشر) ليجيب عن أسئلة العميل حول ماذا (WHAT) يعمل النظام. ويجب أن يكون خالي تماما من أي تفاصيل برمجية أو فنية. والاهم أن يحقق كل المتطلبات التي تم تحديدها سابقا.

التصميم التقني technical design

هذا التصميم سوف يتم تقديمه إلى مطوري النظام ليقوموا هم بتحويله إلى النظام المطلوب، لذا يجب أن يقدم هذا التصميم إجابة شافية لأسئلة المطور عن كيفية (HOW) تطوير النظام. ولمنع إلى تضارب في المفاهيم فإن هذا التصميم عادة ما يكتب باستعمال تعبيرات وأساليب تقنية.

(•·.·´¯`·.·•نهاية الدرس الرابع ولا يوجد نقاش له•·.·´¯`·.·•)

الدرس الخامس: كتابة البرنامج واختباره

أهداف الدرس:

هذا الدرس لن يعلمك لغة برمجة لتكتب بها البرامج، ولكن الهدف منه التعرف على:

- القواعد الصحيحة لكتابة البرامج
- خطة الاختبار وأنواع الاختبارات

الجزء الأول: كتابة البرامج

بعد وضع التصميم للنظام واختيار لغة البرمجة المناسبة، تبدأ الخطوة التي سوف تنقل التصميم المكتوب على الورق إلى واقع. خلال هذا الدرس سوف نناقش أهم القواعد التي على المبرمج إتباعها أثناء كتابة برامجه. ولكن قبل ذلك لنجيب على هذا السؤال الذي لا شك أنه ورد على ذهنك الآن

س: لماذا علينا إتباع هذه القواعد؟

ج: إذا كنت تعمل منفردا في كتابة برامجك، فإن إتباعك لقواعد وأساليب قياسية في البرمجة سوف تساعدك على تنظيم أفكارك لتجنب الوقوع في الأخطاء. كما أنها ستساعدك على اكتشاف أي أخطاء قد تحدث بسرعة وسهولة.

أم إذا كنت تعمل ضمن فريق برمجي، فإن إتباع القواعد والأساليب القياسية في كتابة أجزاء البرامج التي يطلب منك كتابتها، سوف تساعدك وبقية الفريق من تنسيق أعمالكم وتنظيمها، كما أنها ستقلل من عدد الأخطاء في البرنامج وتساعد على اكتشاف ما يقع منها في اسرع وقت ممكن.

تفرض الكثير من شركات البرمجة على مبرمجيها إتباع قواعد قياسية في كتابة برامجهم، وذلك لضمان التكامل في جميع البرامج، كما أن بعض الشركات تعين فرق لاختبار البرامج، غير الفريق الذي قام بالبرمجة ولذلك يجب أن يكون الكود البرمجي مكتوب بطريقة واضحة لجميع من يقرأه، وليس لمن قام بكتابته فقط.

بعض قواعد البرمجة Programming Guidelines

- هياكل التحكم Control Structures

يقصد بها تلك الهياكل التي تتحكم في مسار عمل البرنامج (مثل Goto، if- else) ، وأثناء كتابة هذه الهياكل علنا أن نحاول أن نجعلها واضحة وسهلة التتبع، وخالية من القفزات الواسعة قدر الإمكان. انظر لهذا المثال:

```
benefit = minimum;
      if (age < 75) goto A;
     benefit = maximum;
     goto C;
     if (age < 65) goto B;
     if (age < 55) goto C;
A:    if (age < 65) goto B;
```

```
     benefit = benefit * 1.5 + bonus;
     goto C;
B:    if (age < 55) goto C;
     benefit = benefit * 1.5;
C:   next statement
```

نفس الكود يمكن كتابته على هذا النحو:

```
if (age < 55)  benefit = minimum;
   else if (age < 65)  benefit = minimum + bonus;
   else if (age < 75)  benefit = minimum * 1.5 +bonus;
   else  benefit = maximum;
```

- عالم البرمجة هناك قاعدة تقول أن العمومية ميزةgenerality is a virtue ، لذلك حاول دائما أن تجعل شفراتك البرمجة عامة، لتتمكن من إعادة استعمالها في بقية برامجك بأقل قدر ممكن من التعديل، ولكن حاذر من التمادي في ذلك!
- لا تستخدم أبدا أسماء لا معنى لها لمتغيرات أو بارمترات برنامجك (ينصح بمراجعة هذا الدرس "التسمية في البرنامج، درس لابد من أن يقرأه كل مبرمج!")
- "أريد برنامجا سريعا" وكلنا نريد ذلك، ولكن ما هو الثمن؟!

عندما تفكر في جعل برنامجك أسرع ما يمكن، عليك أن تفكر كذلك في الثمن الذي ستدفعه مقابل ذلك:

1. البرنامج السريع قد يتطلب منك كتابة كود معقد يتطلب منك (ومن فريق العمل) المزيد من الوقت والجهد في كتابته.
2. الوقت الذي تحتاجه عملية اختبار البرنامج المعقد في مختلف حالته.
3. الوقت والجهد الذي تحتاجه لتعديل هذا الكود أو لتطويره.

زمن تنفيذ البرنامج ما هو إلا جزء من معادلة كبيرة لحساب تكلفة البرنامج، لذلك عليك أن تعادل بين السرعة، والجودة، واحتياجات الزبون. ولا تضحي بالبساطة والوضوح من أجل السرعة.

- التوثيق: لا تهمل أبدا توثيق برنامجك، ما سُمي الإنسان إنسانا إلا لنسيانه.

(•.¸¸.•´¯`•.¸¸.نهاية الدرس الخامس - الجزء الأول ولا يوجد نقاش له•.¸¸.•´¯`•.¸¸.•)

الدرس الخامس: كتابة البرنامج واختباره

الجزء الثاني :اختبار البرامج:

وصلنا الآن إلى آخر مرحلة في تطوير النظام، وهي اختبار البرنامج للتأكد من أنه يعمل على النحو الذي يتوقعه الزبون.

قبل تسليم النظام النهائي إلى الزبون تجرى عليه الكثير من الاختبارات، بعضها يعتمد على ما الذي يتم اختباره مثلا:

)أحد مكونات البرنامج - مجموعة من المكونات - جزء من النظام - النظام بالكامل(

والبعض الأخر يعتمد على ما الذي نريد معرفته من هذه الاختبارات، مثلا:

- هل يعمل النظام وفقا لما ورد في المتطلبات؟
- هل يعمل النظام وفقا لما ورد في التصميم؟
- هل يعمل النظام كما يتوقعه الزبون منه؟

مراحل الاختبار:

عند العمل على اختبار نظام من الحجم الكبير، فإن عملية الاختبار تتم على عدة مراحل موجزها في ما يلي:

1. اختبار المكون Module Testing أوcomponent Testing

أول مراحل اختبار النظم، هي اختبار كل مكون على حدى بمعزل عن بقية مكونات النظام، للتأكد من عمله على النحو المتوقع منه. باختبار المعلومات المتحصل عليها (output) منه بعد إمداده بالبيانات اللازمة له.(input)

2. اختبار التكاملIntegration Testing

بعد اختبار كل مكونات النظام والتأكد من سلامة تصميمها، يجب أن نتأكد من أنها ستعمل معا بشكل صحيح وأنه لا يوجد تضارب بين بعضها البعض بحيث أن المعلومات المنتقلة بين هذه المكونات تصل بالهيئة المتوقعة لها. وهذا هو الهدف من اختبار التكامل.

3. اختبار الوظيفةFunction Testing

ويقصد به اختبار النظام بعد تجميع كل مكوناته للتأكد من أنه يؤدي الوظيفة التي يتعين عليه القيام بها، والموضحة في وثائق متطلبات النظام .عندما يجتاز النظام هذا الاختبار يمكننا اعتبار هذا النظام على أنه نظام عاملFunctioning System

4. اختبار الأداء Performance Testing

في هذه الخطوة يتم اختبار أداء البرنامج في بيئة عمل الزبون للتأكد من أن النظام متوافق مع بقية المتطلبات .عند اجتياز النظام لهذا الاختبار يتم التصديق على النظام validated system وبهذا فإننا نعتبر أن النظام أصبح جاهز حسب مفهومنا لما طلبه الزبون.

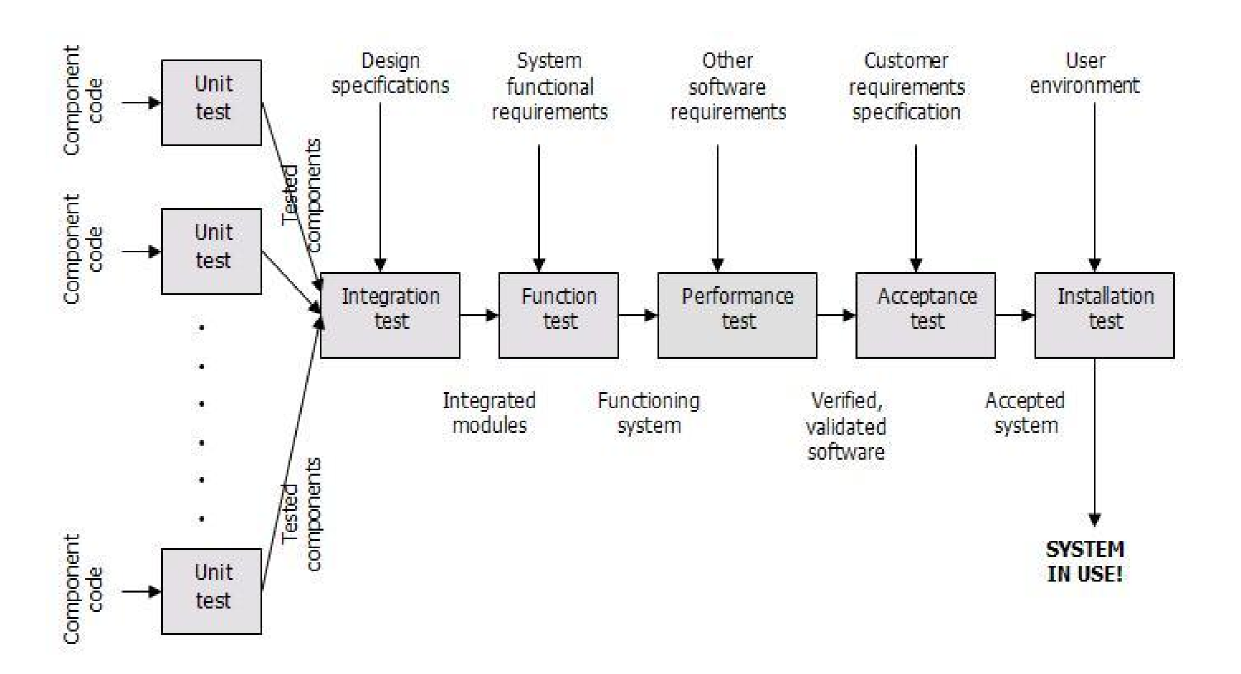
5. اختبار القبول Acceptance Test

يتم إجراء هذا الاختبار للتأكد من أن النظام المحقق موافق لما توقعه الزبون، وبعدها يعد النظام مقبول عند المستخدم والزبون Accepted system

6. اختبار التثبيت Installation Test

الاختبار الأخير يتم فيه تثبيت النظام في بيئة العمل الخاصة به والتأكد من أنه يعمل كما هو مطلوب منه.

الشكل التالي يوضح خطوات تطبيق عملية اختبار النظام، والتي يحسن تطبيقها على اي نظام مهما كان حجمه للتأكد من أنه سيؤدي المهمة المطلوبة منه



(•.¸¸.•´¯`•.¸¸.•نهاية الدرس الخامس - الجزء الثاني ولا يوجد نقاش له•.¸¸.•´¯`•.¸¸.•)

تمت دورة هندسة البرمجيات بحمدالله وتوفيقه ..